2D Projective Geometry

CS 600.361/600.461

Instructor: Greg Hager

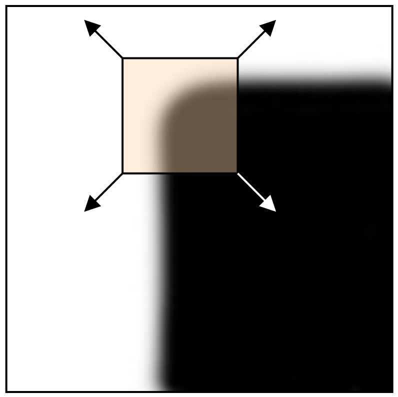
(Adapted from slides by N. Padoy, S. Seitz, K. Grauman and others)

Outline

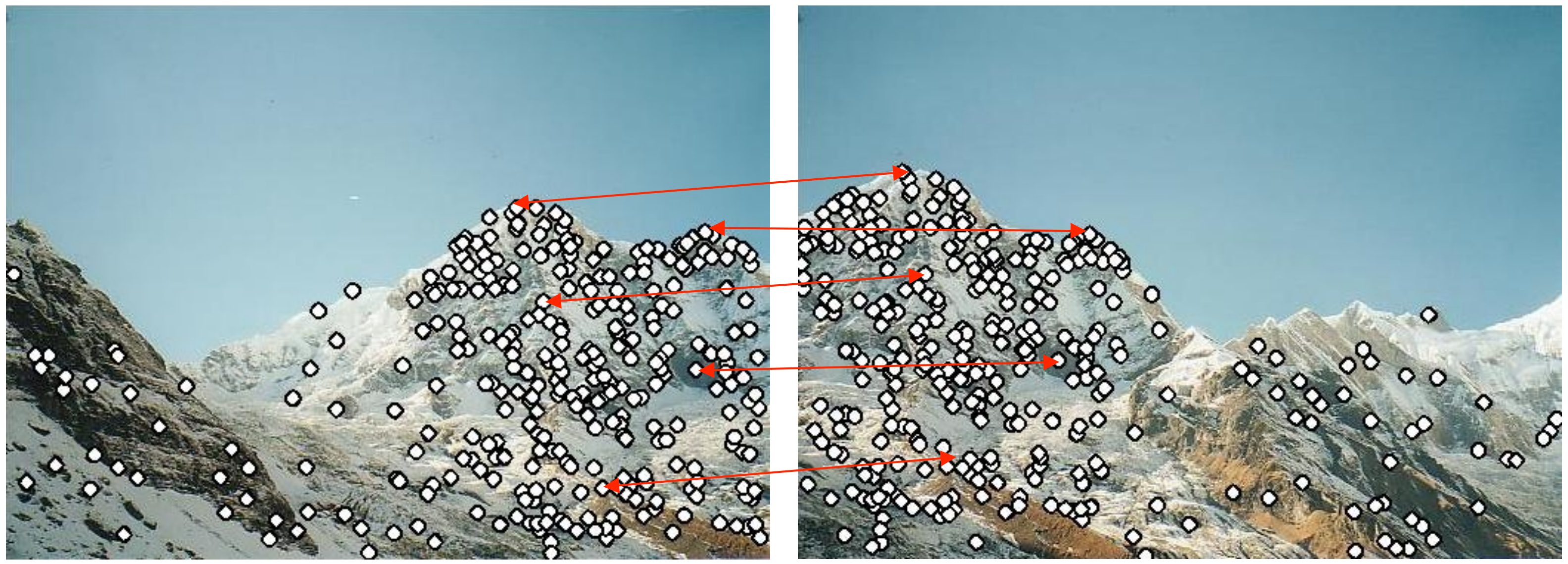
- Linear least squares
- 2D affine alignment
- Image warping
- Perspective alignment
- Direct linear algorithm (DLT)

Reminders

Reminder – Corner detection/matching



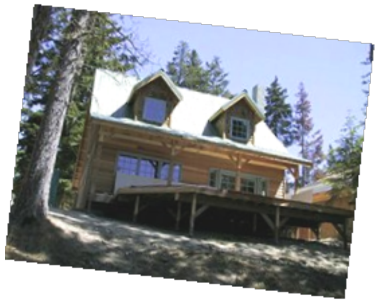
$$M = \sum \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

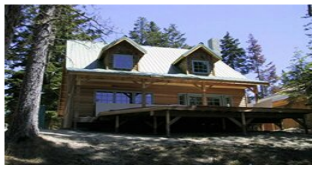


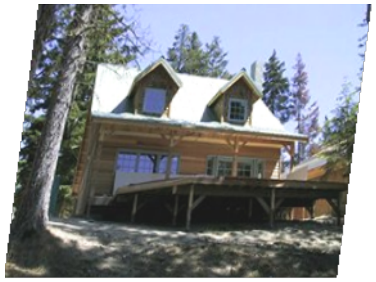
Reminder – Parametric (global) warping

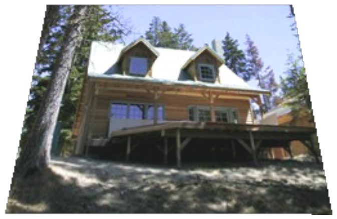
Examples of parametric warps:

translation


rotation


aspect


affine


perspective

Source: Alyosha Efros

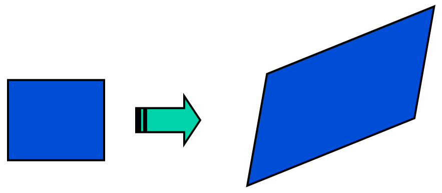
Reminder – 2D Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine transformations are combinations of …

- Linear transformations, and
- Translations

Parallel lines remain parallel

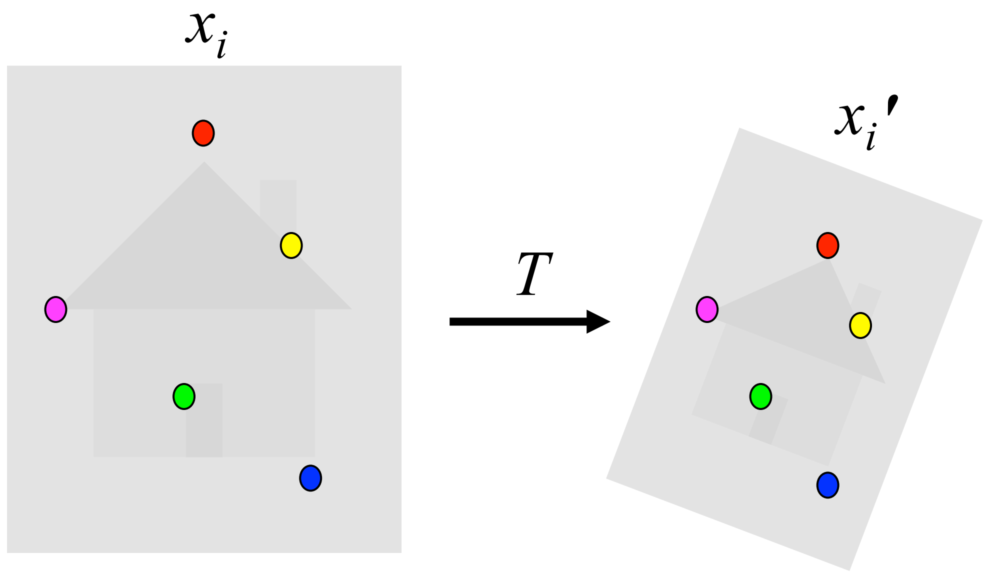


Reminder – Alignment problem

We have previously considered how to **fit a model to image evidence**

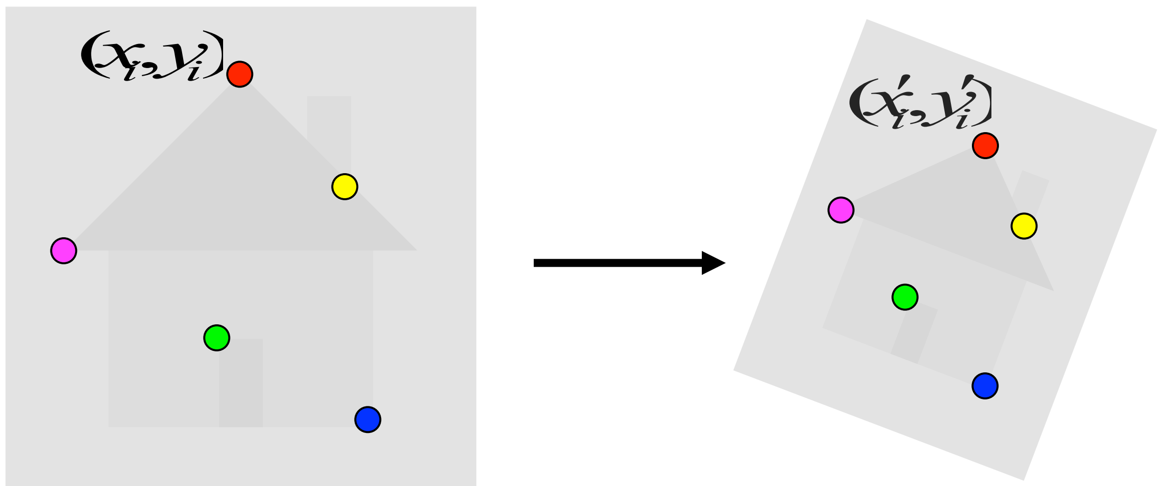
- e.g., a line to edge points

In alignment, we will **fit the parameters of some transformation** according to a set of matching feature pairs ("correspondences").



Reminder – Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Reminder – Fitting an affine transformation

- Least square minimization:

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

$$\min \left\| Ax - b \right\|^2$$

Reminder – Singular Value Decomposition

Given any *m*×*n* real matrix **A**, algorithm to find matrices **U**, **V**, and **D** such that

A = **U D V**^{T}

U is *m*×m and orthogonal

D is m×*n* and diagonal

V is *n*×*n* and orthogonal

$$d_1 \geq d_2 \geq \cdots \geq d_p \geq 0 \quad \text{for p=min(m,n)}$$

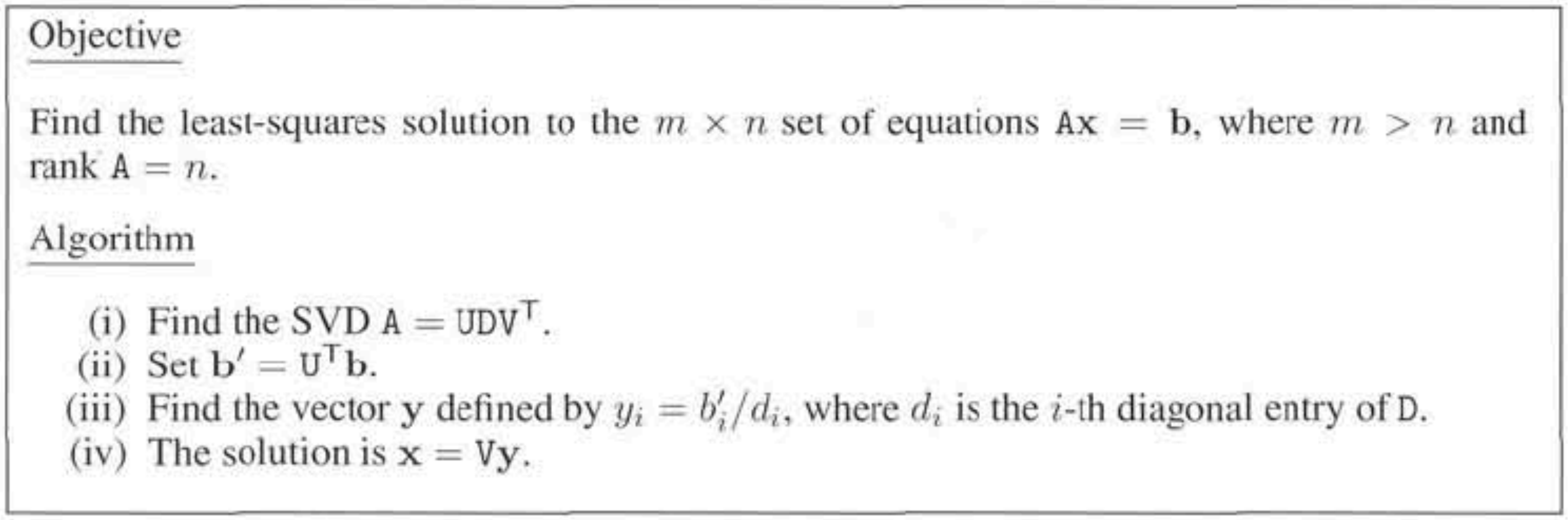
$$\begin{pmatrix} \\ \\ \mathbf{A} \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ \mathbf{U} \\ \\ \\ \end{pmatrix} \begin{pmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_p \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \\ \mathbf{V} \\ \\ \end{pmatrix}^{T}$$

Linear least squares

(Board)

LLS - Method 1

Linear least-squares solution to an overdetermined full-rank set of linear equations



<u>Objective</u>

Find the least-squares solution to the $m \times n$ set of equations $\mathtt{A}\mathbf{x} = \mathbf{b}$, where $m > n$ and rank $\mathtt{A} = n$.

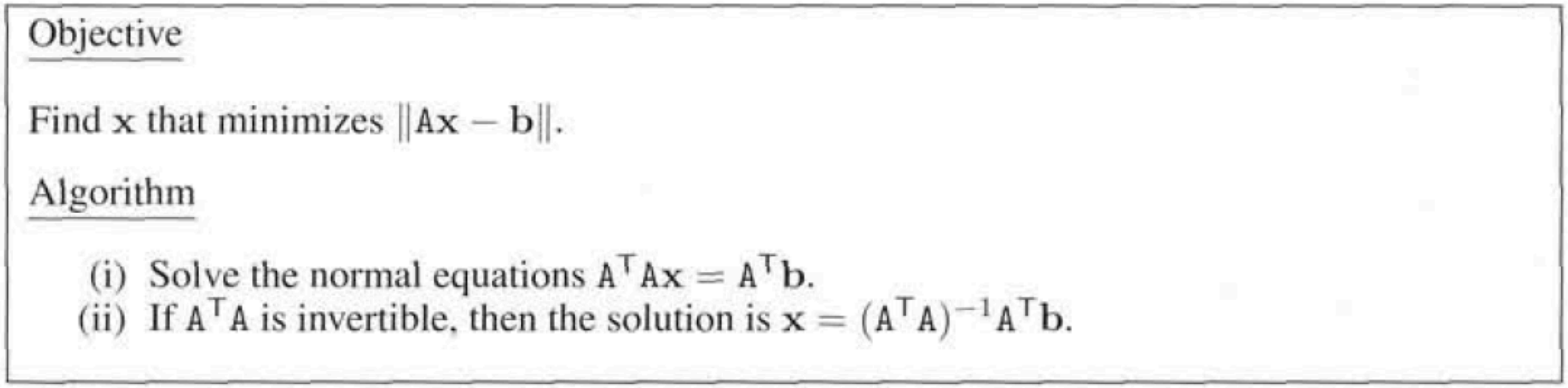
<u>Algorithm</u>

(i) Find the SVD $\mathtt{A} = \mathtt{UDV}^\mathsf{T}$.
(ii) Set $\mathbf{b}' = \mathtt{U}^\mathsf{T}\mathbf{b}$.
(iii) Find the vector $\mathbf{y}$ defined by $y_i = b'_i/d_i$, where d_i is the i-th diagonal entry of $\mathtt{D}$.
(iv) The solution is $\mathbf{x} = \mathtt{V}\mathbf{y}$.

(Picture from Hartley's book, appendix 5)

LLS – Method 2

Linear least-squares solution to an overdetermined full-rank set of linear equations



Objective

Find $\mathbf{x}$ that minimizes $\|\mathtt{A}\mathbf{x} - \mathbf{b}\|$.

Algorithm

(i) Solve the normal equations $\mathtt{A}^{\mathsf{T}}\mathtt{A}\mathbf{x} = \mathtt{A}^{\mathsf{T}}\mathbf{b}$.
(ii) If $\mathtt{A}^{\mathsf{T}}\mathtt{A}$ is invertible, then the solution is $\mathbf{x} = (\mathtt{A}^{\mathsf{T}}\mathtt{A})^{-1}\mathtt{A}^{\mathsf{T}}\mathbf{b}$.

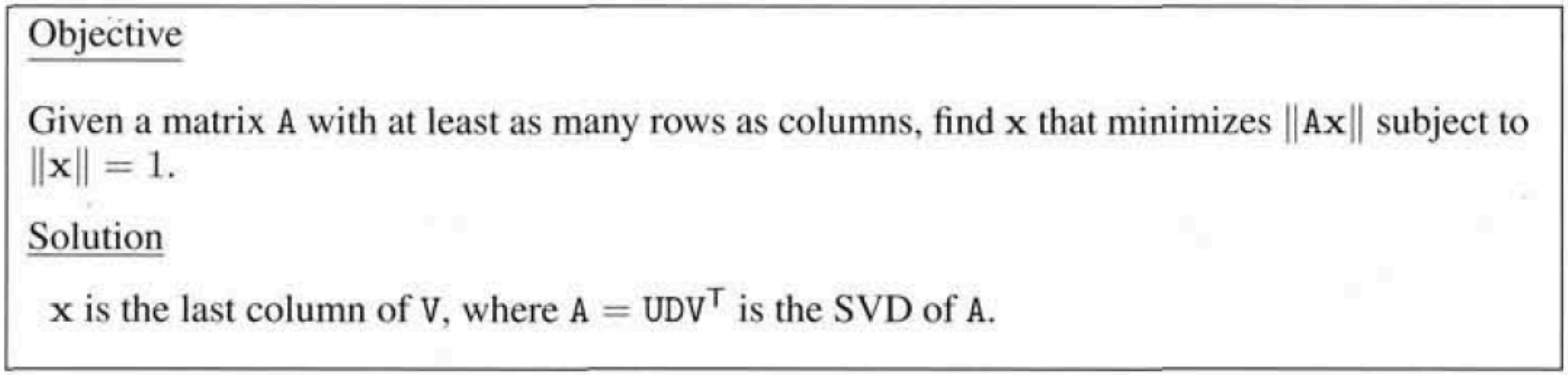
Matlab: check the functions *svd, pinv, mldivide*

Note: if A is not full rank, this is in general a different solution than the one from method 1

(Picture from Hartley's book, appendix 5)

LLS – Method 3

Linear least-squares solution to a homogeneous system of linear equations



Objective

Given a matrix A with at least as many rows as columns, find $\mathbf{x}$ that minimizes $\|A\mathbf{x}\|$ subject to $\|\mathbf{x}\| = 1$.

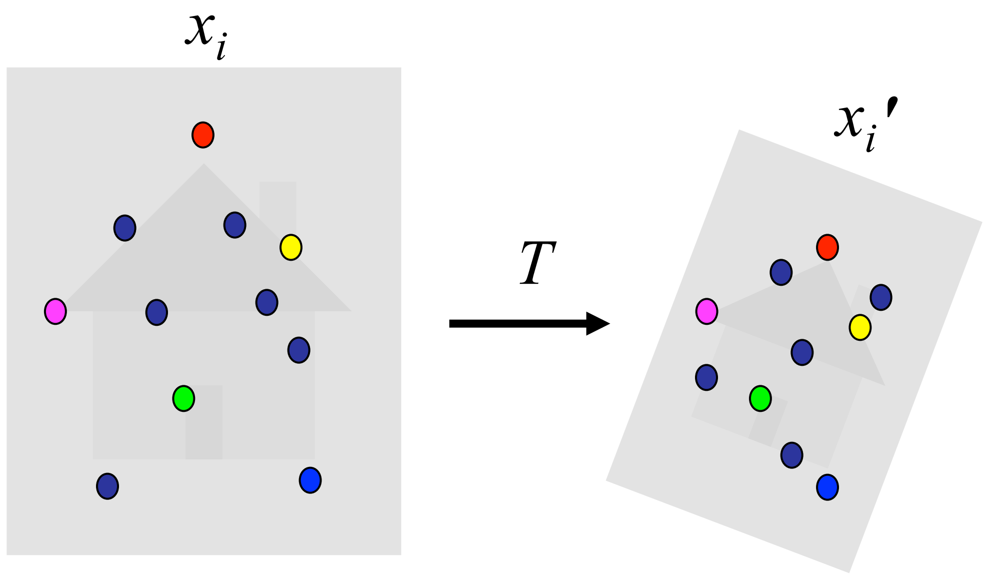
Solution

$\mathbf{x}$ is the last column of V, where $A = UDV^\top$ is the SVD of A.

(Picture from Hartley's book, appendix 5)

RANSAC for affine alignment

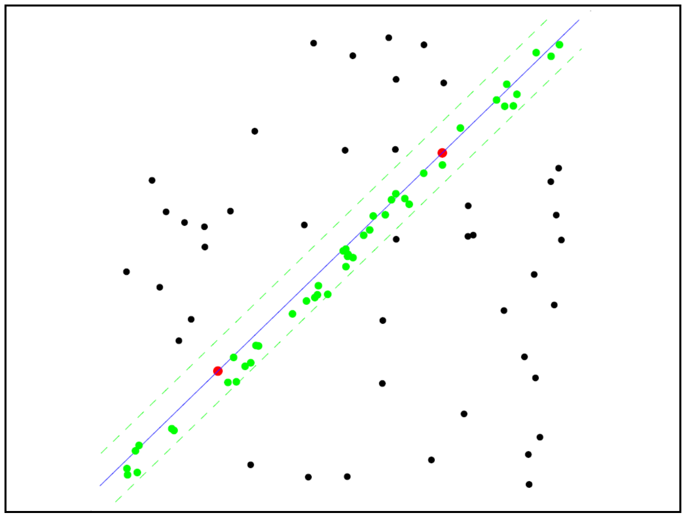
How to deal with noisy correspondences ?



Reminder: RANSAC for line fitting

Repeat ***N*** times:

- Draw ***s* points** uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers



Lana Lazebnik

RANSAC for affine alignment

Repeat ***N*** times:

- Draw ***s*** point **correspondences** uniformly at random
- Fit **affine transformation T** to these ***s*** correspondences
- Find **inliers** to this transformation T among the remaining correspondences
- If there are ***d*** or more inliers, accept T and refit using all inliers

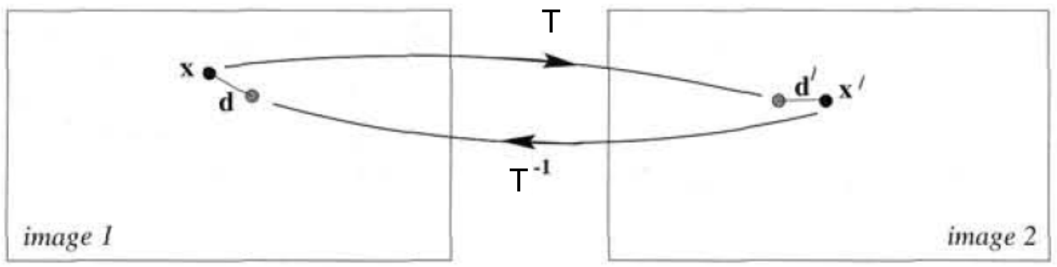
What is in **inlier** to a transformation T ?

Inliers to a transformation T

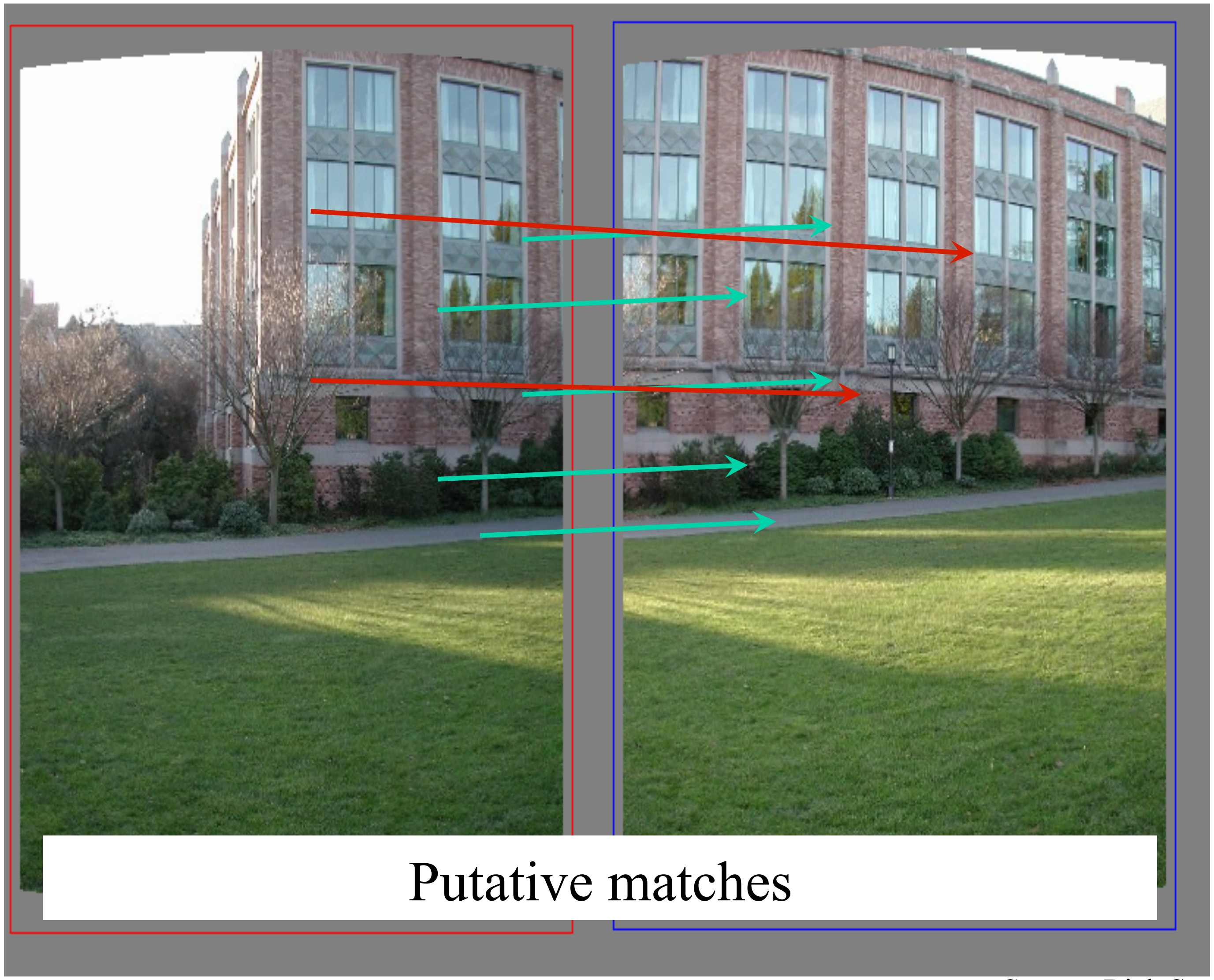
- Two of several possibilities
  - Using asymmetric transfer error

    (x,x') such that ||x'-Tx|| below a threshold ***t***
  - Using symmetric transfer error

    (x,x') such that ||x'-Tx||+||x-inv(T)x'|| below ***t***

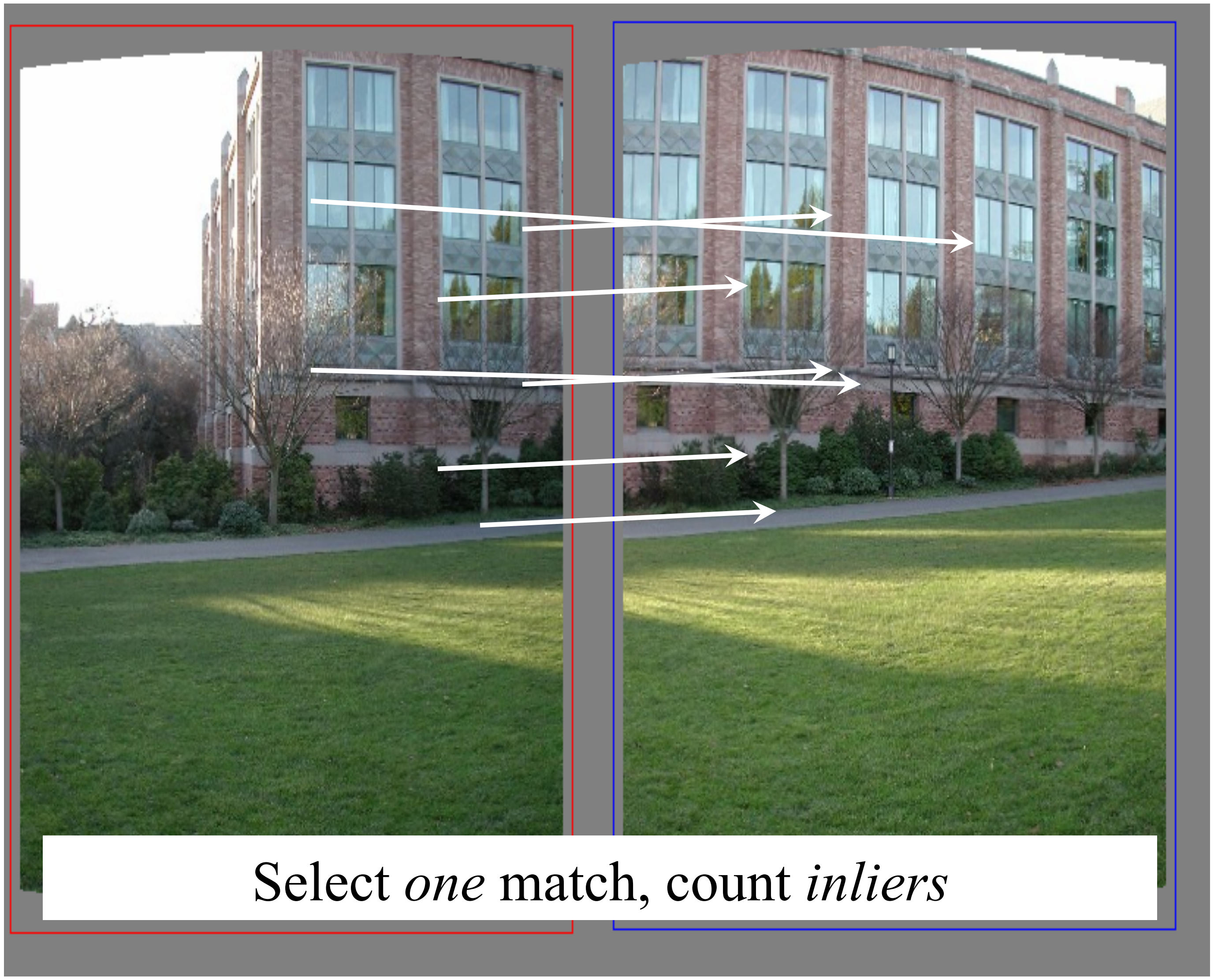


RANSAC example: Translation

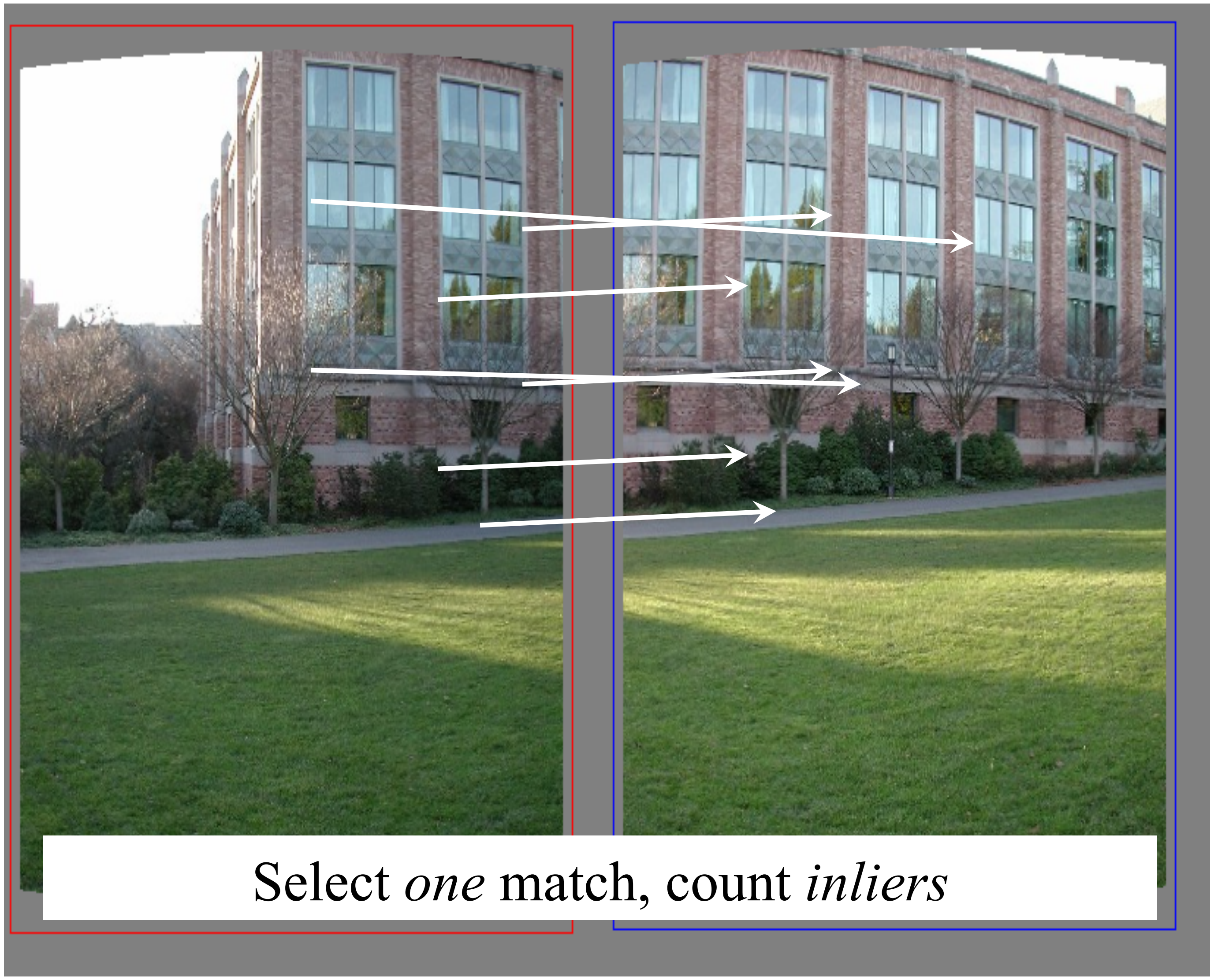


Source: Rick Szeliski

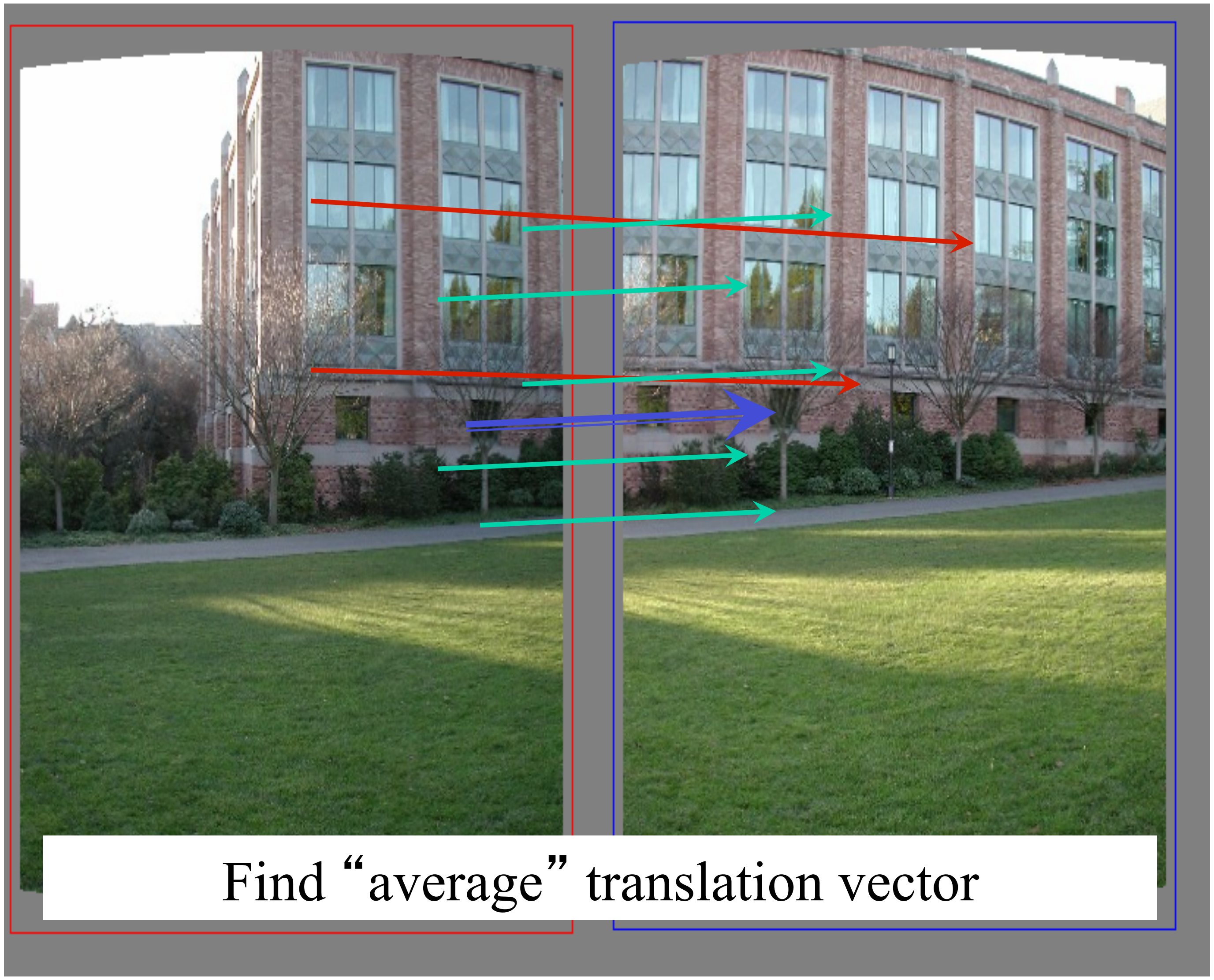
RANSAC example: Translation


Select *one* match, count *inliers*

RANSAC example: Translation


Select *one* match, count *inliers*

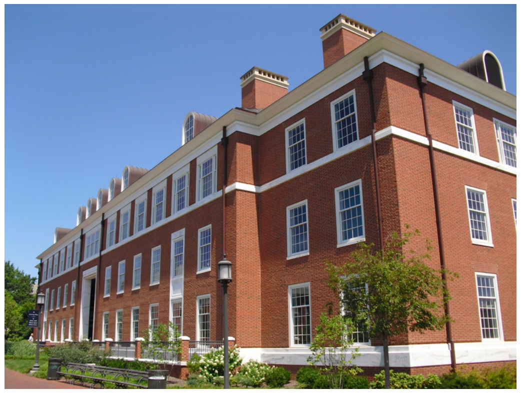
RANSAC example: Translation



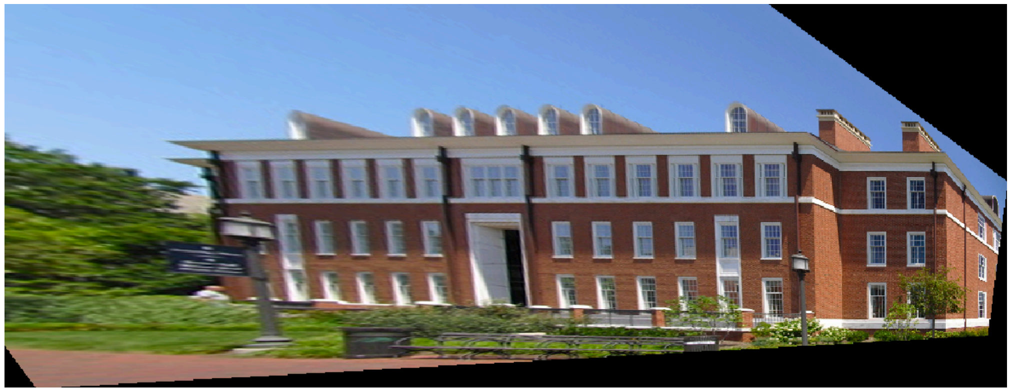
Find “average” translation vector

Note

Is this an affine transformation ?



$\xrightarrow{T}$

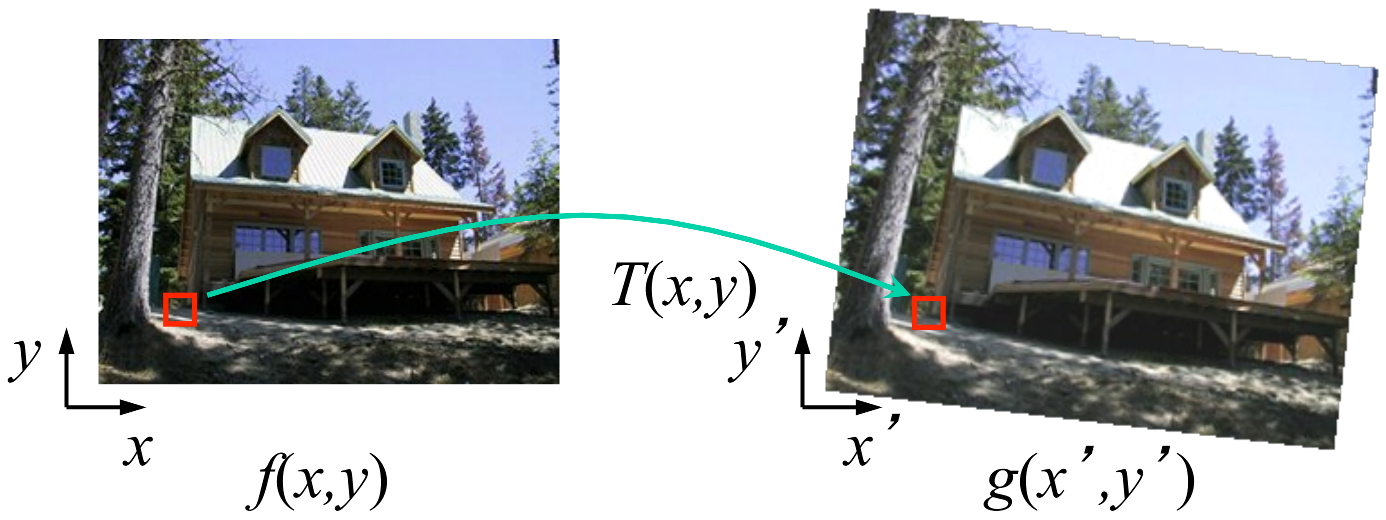


We are not fully done yet…

- How do we correctly warp (or unwarp) an image, per pixel, knowing T ?
- What about the case of 2D perspective transformations ?

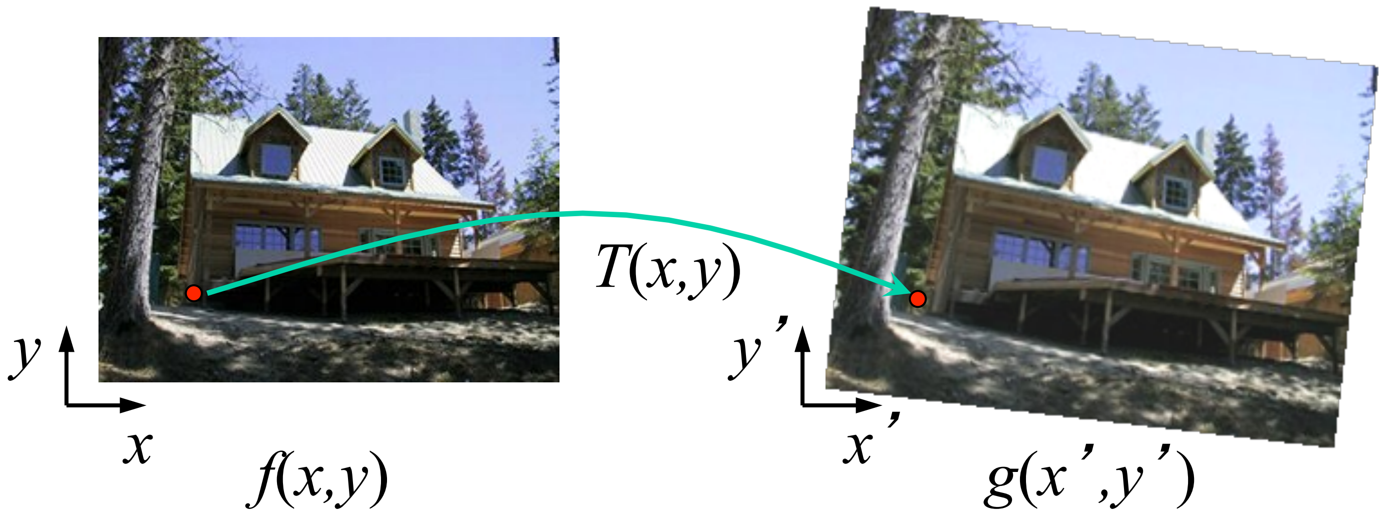
Image warping

Image warping



Given a coordinate transform and a source image *f*(*x,y*), how do we compute a transformed image *g(x',y')* = *f*(*T*(*x,y*))?

Forward warping

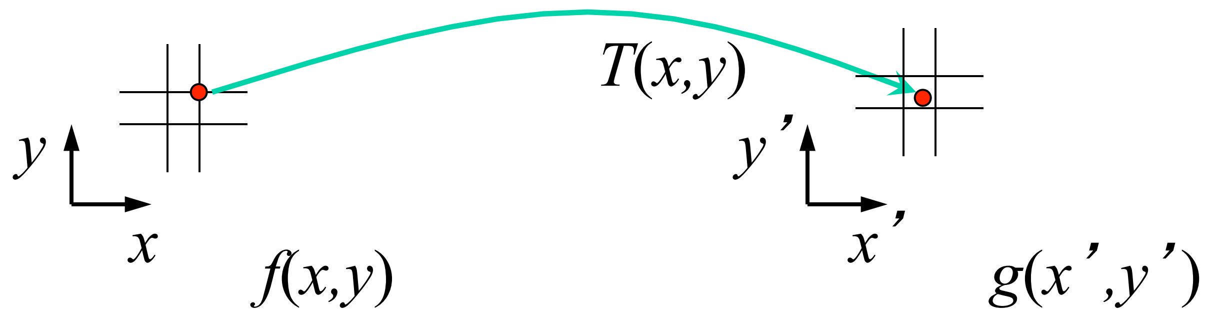


Send each pixel $f(x,y)$ to its corresponding location
$(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

Slide from Alyosha Efros, CMU

Forward warping



Send each pixel $f(x,y)$ to its corresponding location
$(x',y') = T(x,y)$ in the second image

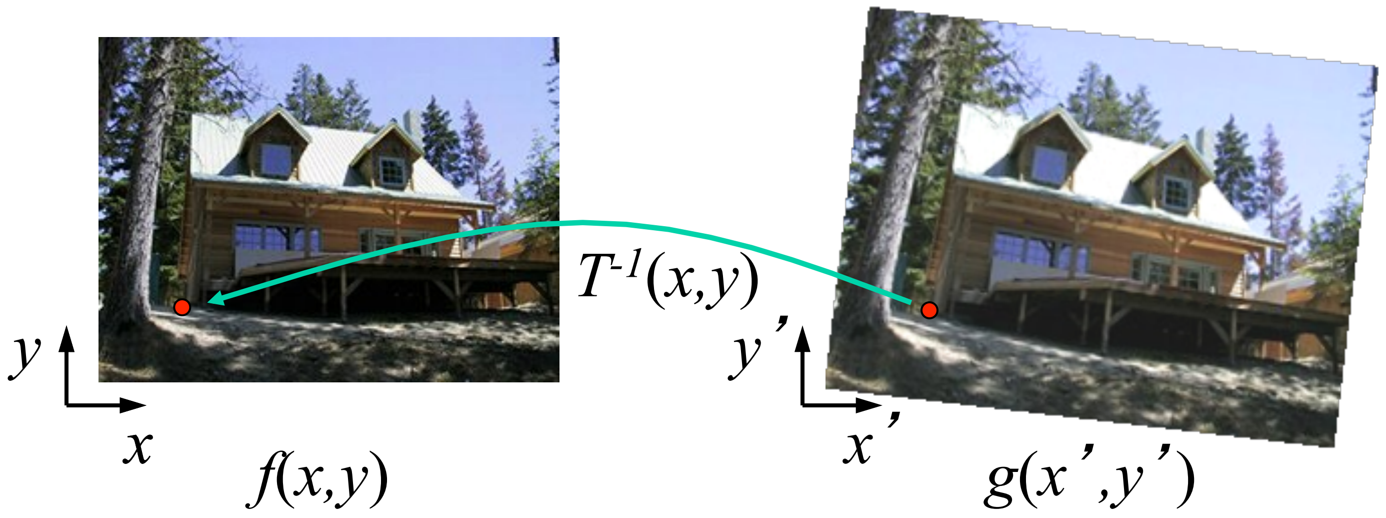
Q: what if pixel lands "between" two pixels?

A: distribute color among neighboring pixels (x',y')

- Known as "splatting"

Slide from Alyosha Efros, CMU

Inverse warping



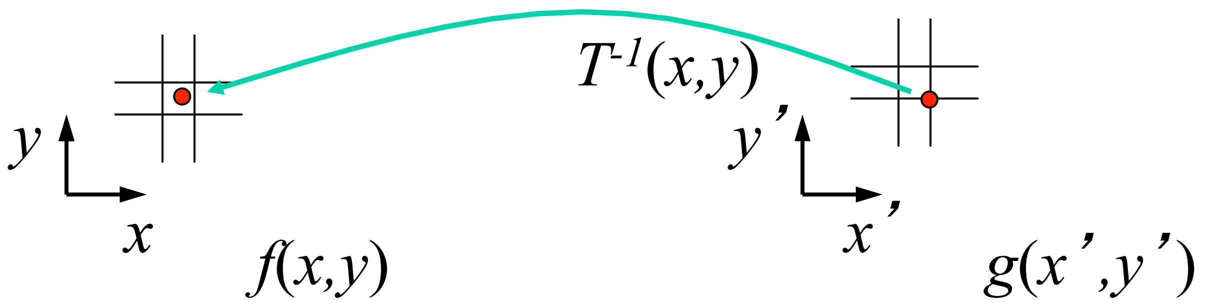
Get each pixel $g(x',y')$ from its corresponding location

$(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from “between” two pixels?

Slide from Alyosha Efros, CMU

Inverse warping



Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

A: *Interpolate* color value from neighbors

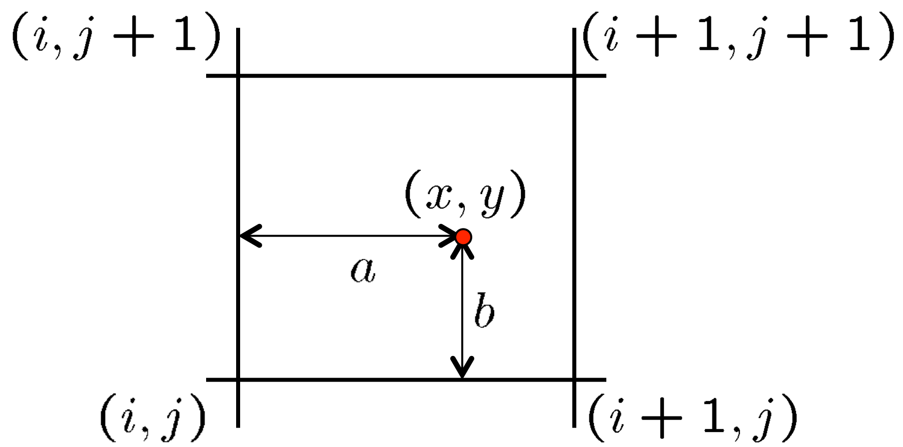
– nearest neighbor, bilinear…

```
>> help interp2
```

Slide from Alyosha Efros, CMU

Bilinear interpolation

Sampling at *f(x,y):*



$$\begin{aligned} f(x,y) = \; & (1-a)(1-b) && f[i,j] \\ & +a(1-b) && f[i+1,j] \\ & +ab && f[i+1,j+1] \\ & +(1-a)b && f[i,j+1] \end{aligned}$$

Slide from Alyosha Efros, CMU

2D projective geometry

Projective geometry

- 2D projective geometry
  - Points on a plane (projective plane $\mathcal{P}^2$) are represented in homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Objective: study projective transformations and their invariants

- Definition: a **projective transformation h** is an invertible mapping from $\mathcal{P}^2$ to $\mathcal{P}^2$ that preserves collinearity between points *(x1, x2, x3 on same line => h(x1), h(x2),h(x3) on same line)*

- projective transformation = homography = collineation

Homography

Theorem

A mapping $\mathbf{h} : \mathcal{P}^2 \rightarrow \mathcal{P}^2$ if a projective transformation if and on if there exists an invertible 3x3 matrix H such that for any point **x** represented in homogeneous coordinates, h(**x**)=H**x**

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

x'=h(**x**) H **x**

Note: equation is up to a scale factor

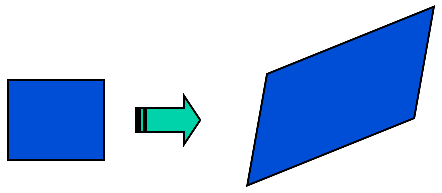
Reminder – 2D Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine transformations are combinations of …

- Linear transformations, and
- Translations

Parallel lines remain parallel



2D Projective Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Parallel lines do not necessarily remain parallel

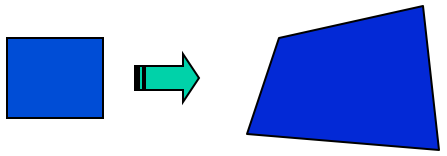
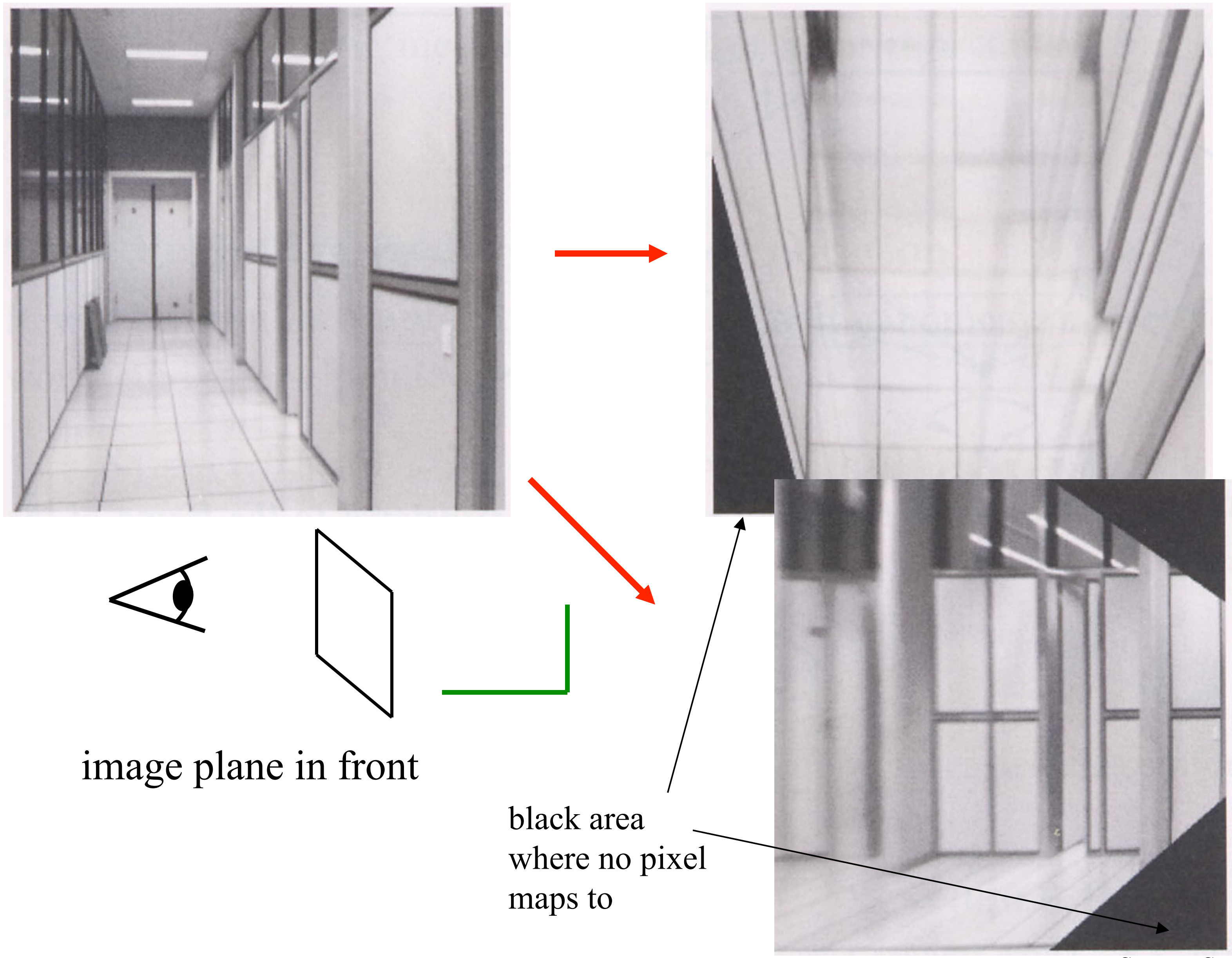


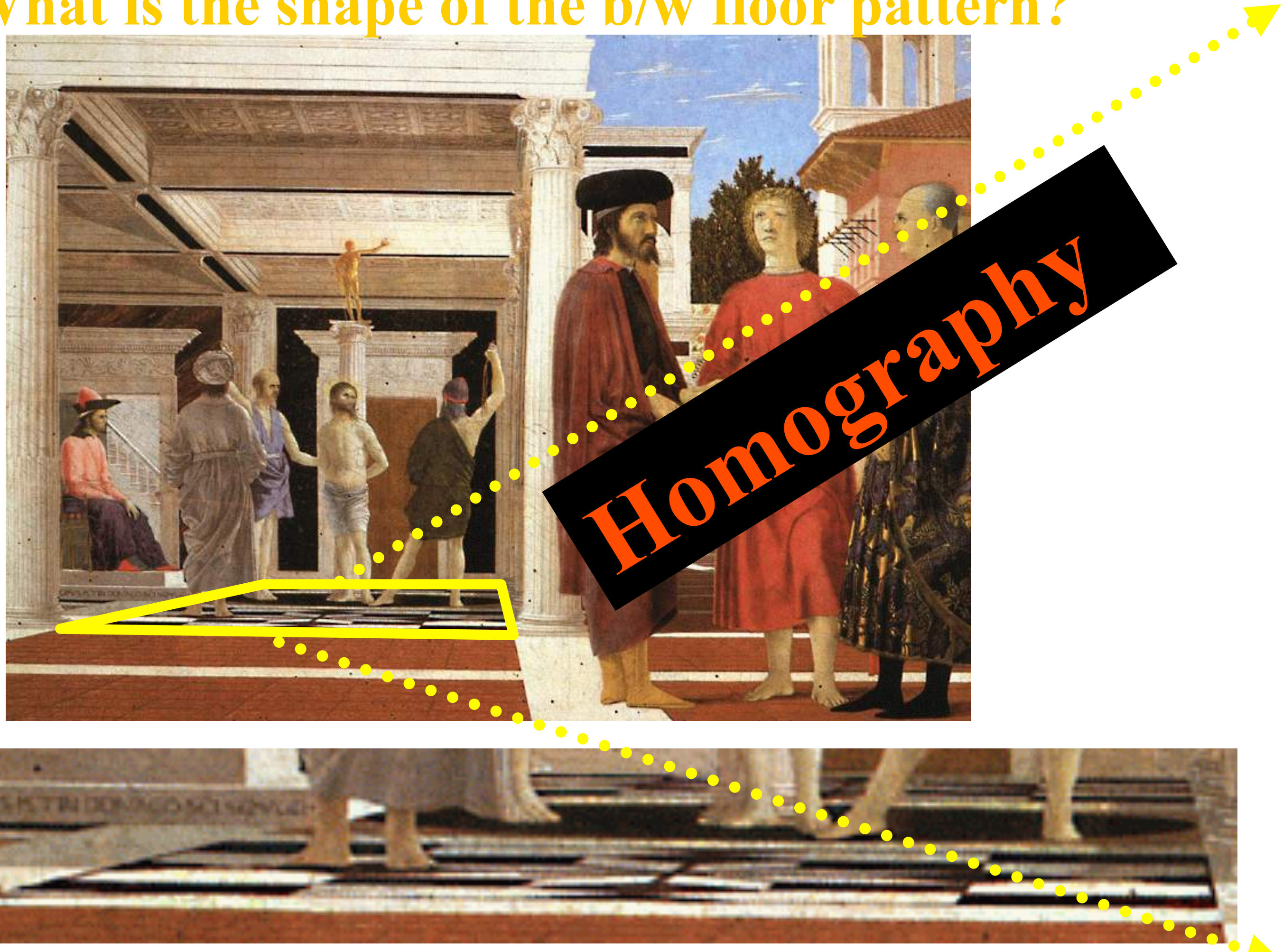
Image warping with homographies



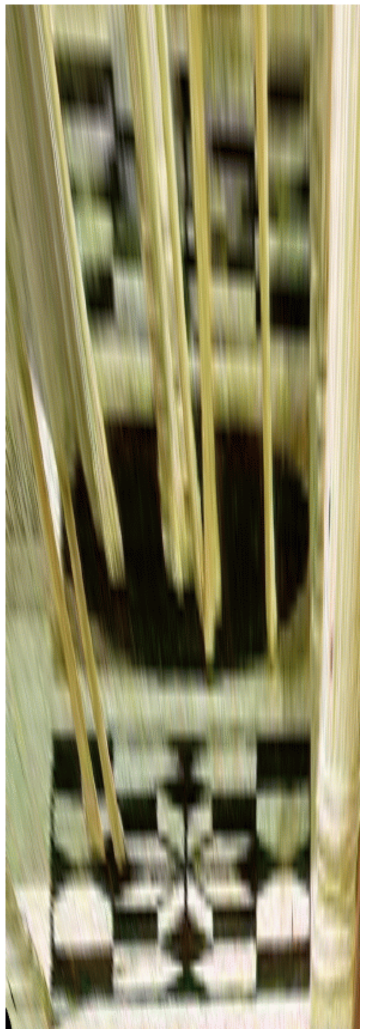
Source: Steve Seitz

Analysing patterns and shapes

What is the shape of the b/w floor pattern?



The floor (enlarged)

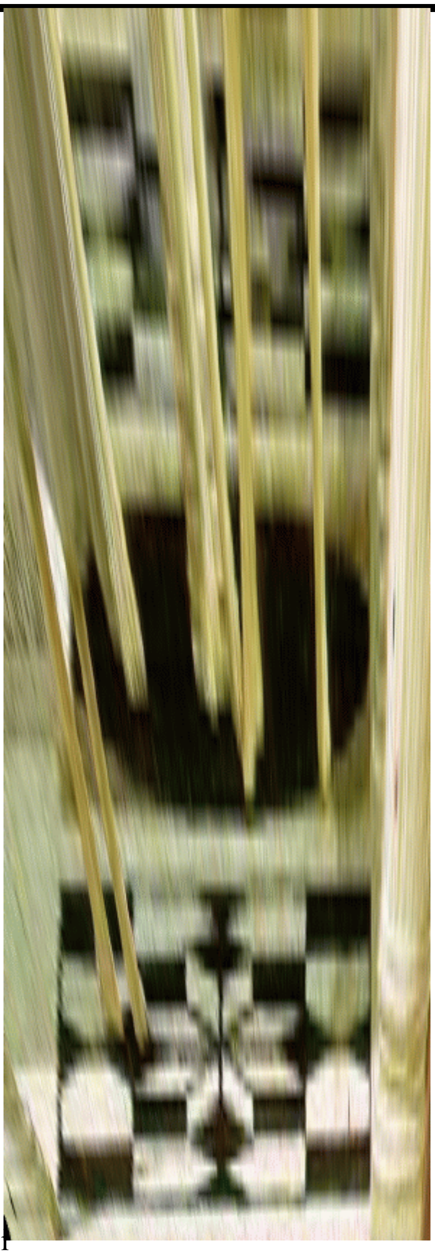


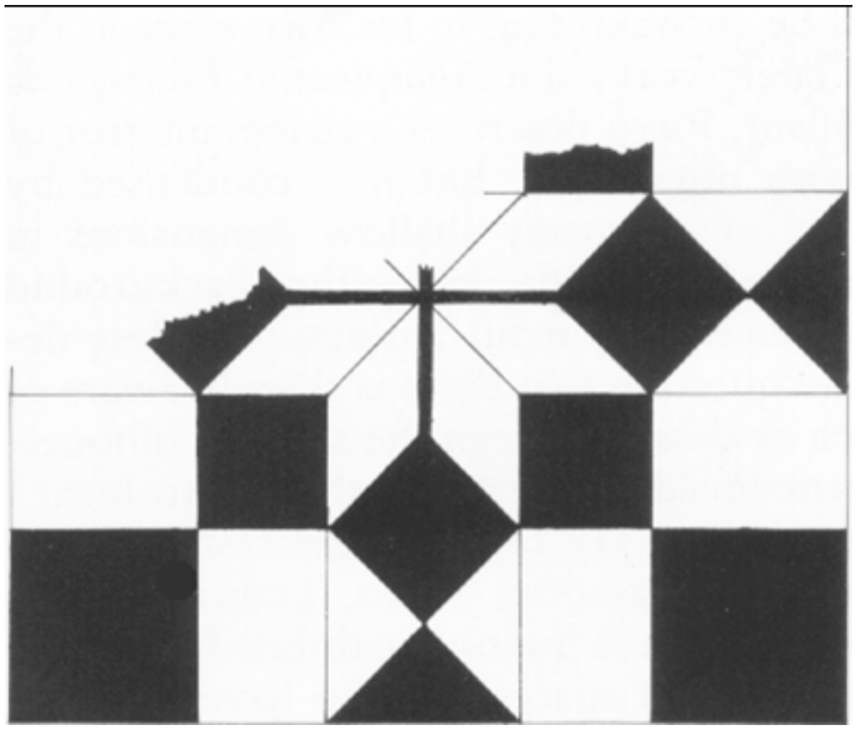
Automatically rectified floor

Slide from Antonio Criminisi

Analysing patterns and shapes

Automatic rectification

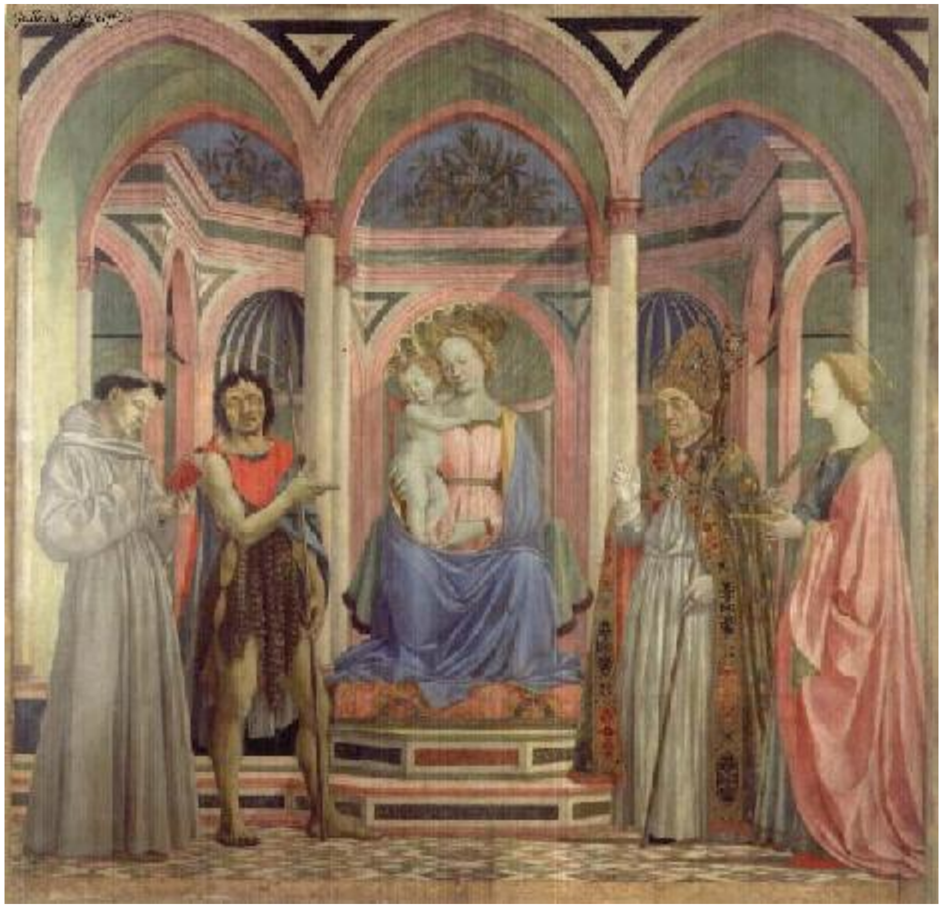




From Martin Kemp *The Science of Art*
(manual reconstruction)

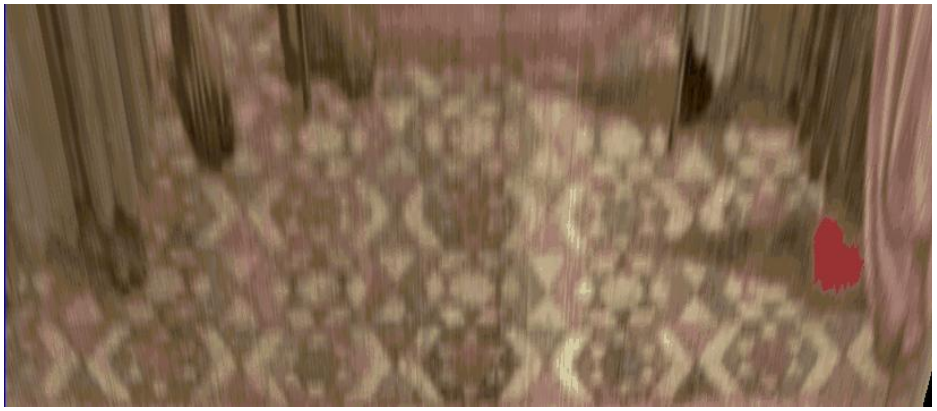
Slide from Antonio Criminisi

Analysing patterns and shapes



***St. Lucy Altarpiece,* D. Veneziano**

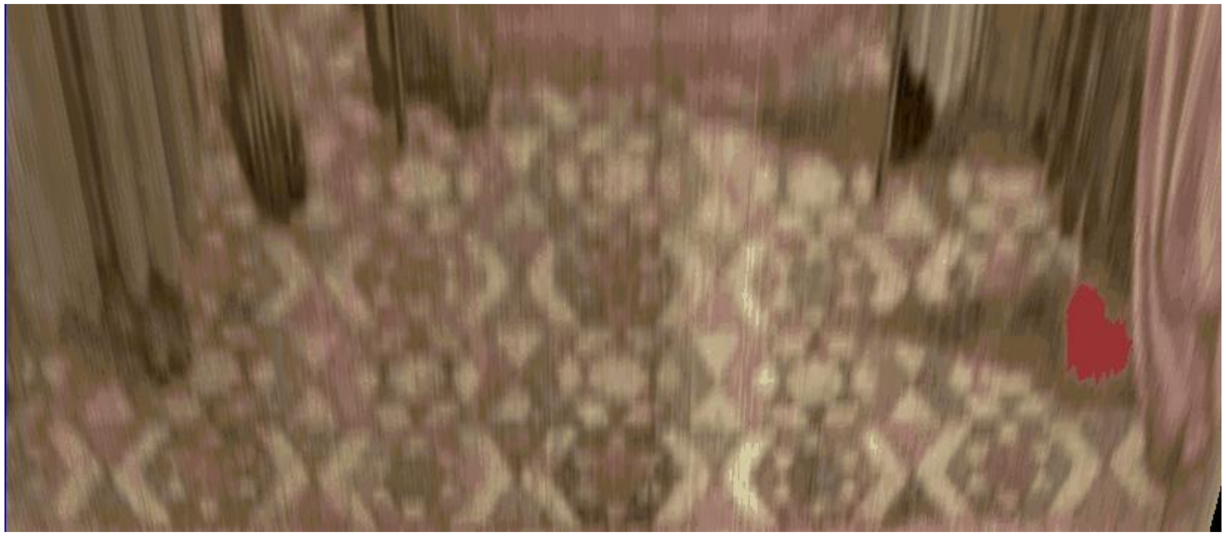
What is the (complicated) shape of the floor pattern?



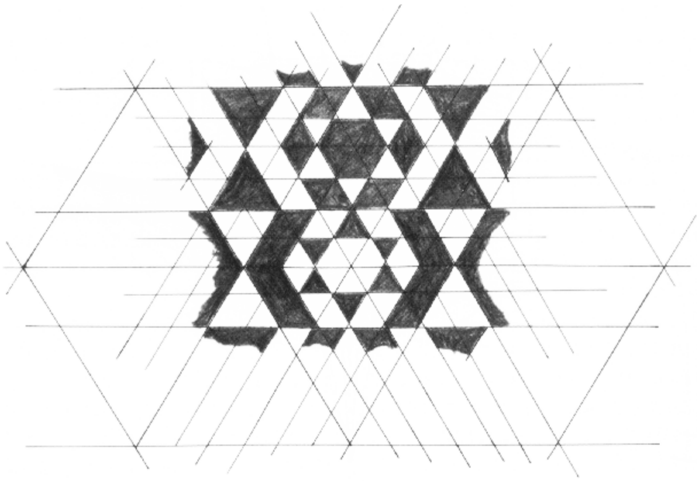
Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes



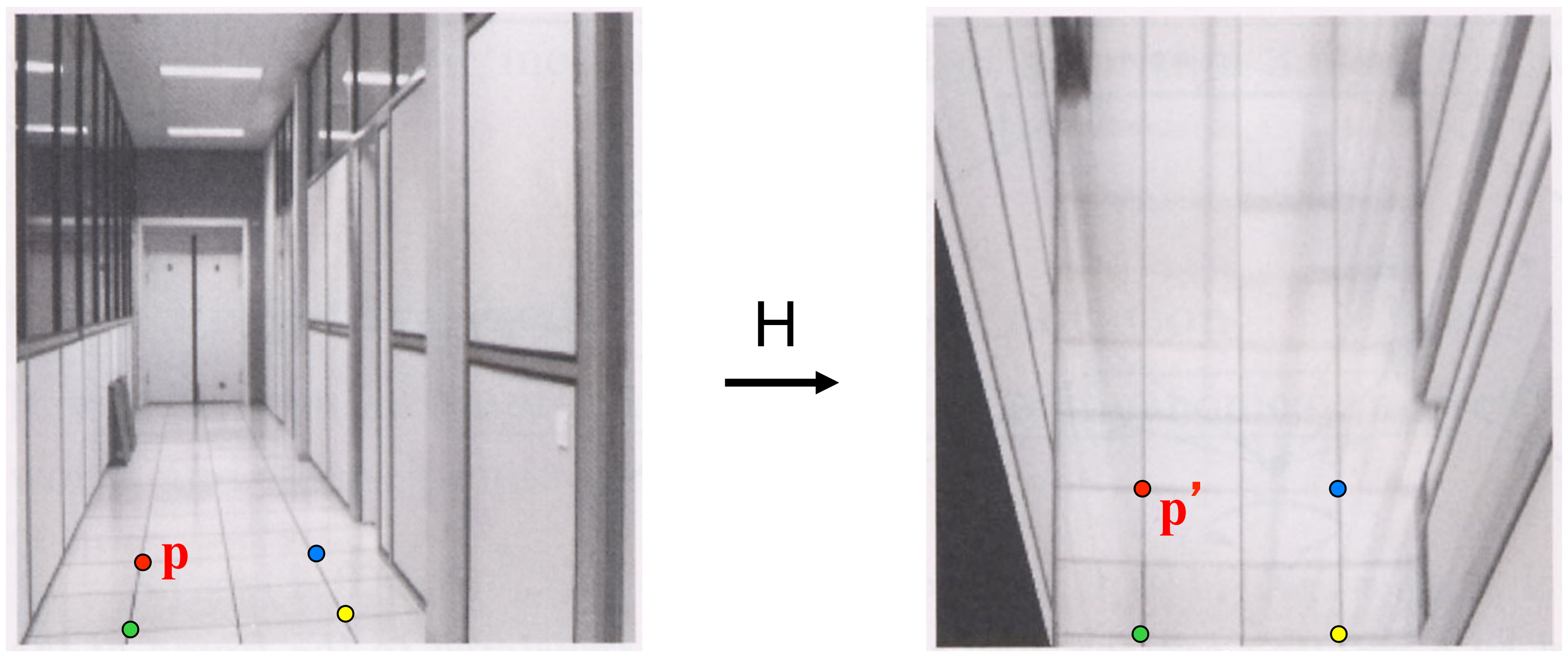
Automatic rectification



From Martin Kemp, *The Science of Art*
(manual reconstruction)

Slide from Criminisi

Image rectification



How do we compute H ?

Solving for homographies

$$\mathbf{p'} = \mathbf{Hp}$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ \mathrm{w} \end{bmatrix}$$

- Up to scale. So, there are 8 degrees of freedom (DoF).
- Set up a system of linear equations:

$$\mathbf{Ah = 0}$$

where vector of unknowns h = [h1,h2,h3,h4,h5,h6,h7,h8,h9]^T

- Need at least 8 eqs, but the more the better…
- Solve using least-squares

(BOARD)

Summary: DLT algorithm

<u>Objective</u>

Given n≥4 2D to 2D point correspondences $\{x_i \leftrightarrow x_i'\}$, determine the 2D homography matrix H such that $x_i' = Hx_i$

<u>Algorithm</u>

(i) For each correspondence $x_i \leftrightarrow x_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble *n* 2x9 matrices A_i into a single 2*n*x9 matrix A

(iii) Obtain SVD of A. Solution for h is last column of V

(iv) Determine H from h

$$X_i = \begin{bmatrix} x_i & y_i & w_i \end{bmatrix}^T$$

$$X_i' = \begin{bmatrix} x_i' & y_i' & w_i' \end{bmatrix}^T$$

$$\begin{bmatrix} 0 & 0 & 0 & -w_i'x_i & -w_i'y_i & -w_i'w_i & y_i'x_i & y_i'y_i & y_i'w_i \\ w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \\ -y_i'x_i & -y_i'y_i & -y_i'w_i & x_i'x_i & x_i'y_i & x_i'w_i & 0 & 0 & 0 \end{bmatrix}$$

Conclusion

- Today
  - Affine alignment
  - RANSAC in presence of outliers
  - Image warping
  - Homography

- Next time
  - More on homography estimation
  - Mosaicing
  - More projective geometry